BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

G06K15/00

Office européen des brevets

(11) Publication number : 0 470 782 A3

(12) EUROPEAN PATENT APPLICATION

(21) Application number : 91307141.1

(51) Int. Cl.[5] : G06K 15/00

(22) Date of filing : 02.08.91

(30) Priority : 08.08.90 US 564417

(43) Date of publication of application :
12.02.92 Bulletin 92/07

(84) Designated Contracting States :
AT BE CH DE DK ES FR GB IT LI NL SE

(88) Date of deferred publication of search report :
14.10.92 Bulletin 92/42

(71) Applicant : THE PEERLESS GROUP
2629 Manhattan Beach Boulevard
Redondo Beach, California 90278 (US)

(72) Inventor : Butterfield, Stephen R.
1515 11th Street
Manhattan Beach, California 90266 (US)
Inventor : Phillips, Donald E.
8793 Brookdale Drive
Garden Grove, California 92644 (US)
Inventor : Renshaw, Barbara B.
829 Pacific Avenue
Manhattan Beach, California 90266 (US)
Inventor : Nelson, Steven K.
948 West 245th Street
Harbor City, California 90710 (US)
Inventor : Hossley, Robert F.
622 Loma Vista
El Segundo, California 90245 (US)

(74) Representative : Wombwell, Francis
Potts, Kerr & Co. 15, Hamilton Square
Birkenhead Merseyside L41 6BR (GB)

(54) Image rendering process and apparatus.

(57) An apparatus for creating a page image which may include complex graphics information for display on a continuous synchronous raster output device by generating bands of graphics information which use an amount of memory which is less than the amount which would be needed to store a complete page image, said apparatus comprising :

a) means for receiving and interpretting commands in a page description language which define the page image to be output ;

b) means for generating a set of graphics orders from the interpretted page description language commands such that :

i) said graphics orders can be processed into bands of bitmap images which can delivered to the output device at a speed required by the output device ; and

ii) while the graphics information in a first one of said bands is being output by said output device, the bitmap images for a second one of said bands is being generated ;

c) means for generating said bands of bitmap images from said graphics orders ;

d) means for outputting said bands of bitmap images to said output device.

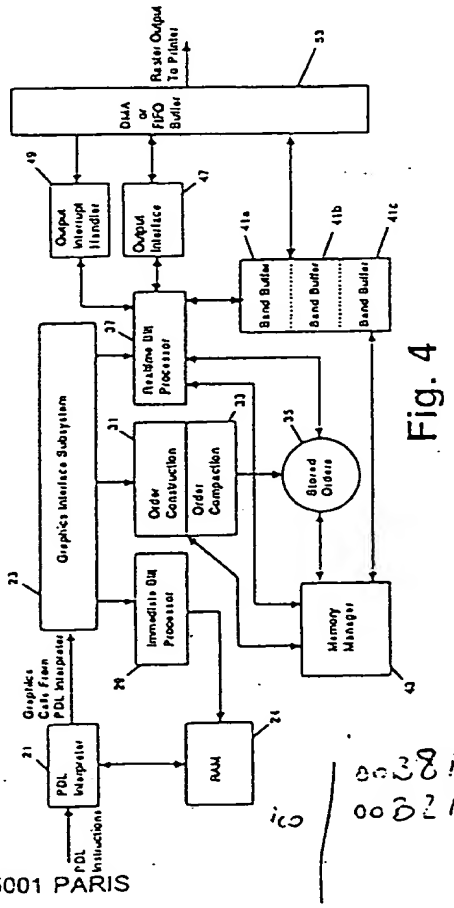

Fig. 4

EP 0 470 782 A3

1 graphic image stored and processed in bands; real time conversion to bitmap while printing

European Patent Office

EUROPEAN SEARCH REPORT

Application Number

EP 91 30 7141

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
Y,P	EP-A-0 428 370 (CANON KABUSHIKI KAISHA) * the whole document *	1	G06K15/00
Y	EP-A-0 283 157 (NATIONAL BUSINESS SYSTEM) * abstract * * column 4, line 1 - column 5, line 49 * * column 15, line 26 - column 16, line 5 *	1	
A	US-A-4 703 515 (BARODDY) * abstract; claims 1-4; figures 1-3 * * column 1, line 5 - column 65 * * column 2, line 34 - column 5, line 10 *	1-2,4	
A	IBM TECHNICAL DISCLOSURE BULLETIN. vol. 31, no. 10, March 1989, NEW YORK US pages 352 - 354; ARMONK: 'MEMORY ALLOCATION AND DEALLOCATION FOR PRINT DATA IN APA PRINTERS' * the whole document *	1	

TECHNICAL FIELDS SEARCHED (Int. Cl.5)

G06K
G09G

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	19 AUGUST 1992	BEAUCE C.Y.C.

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P0401)

EP 0 470 782 (1)
G06K15/00

-1- * -

G06K15/00

(11) Publication number : 0 470 782 A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number : 91307141.1

(51) Int. Cl.[5] : G06K 15/00

(22) Date of filing : 02.08.91

(30) Priority : 08.08.90 US 564417

(43) Date of publication of application :
12.02.92 Bulletin 92/07

(84) Designated Contracting States :
AT BE CH DE DK ES FR GB IT LI NL SE

(71) Applicant : THE PEERLESS GROUP
2629 Manhattan Beach Boulevard
Redondo Beach, California 90278 (US)

(72) Inventor : Butterfield, Stephen R.
1515 11th Street
Manhattan Beach, California 90266 (US)

Inventor : Phillips, Donald E.
8793 Brookdale Drive
Garden Grove, California 92644 (US)
Inventor : Renshaw, Barbara B.
829 Pacific Avenue
Manhattan Beach, California 90266 (US)
Inventor : Nelson, Steven K.
948 West 245th Street
Harbor City, California 90710 (US)
Inventor : Hossley, Robert F.
622 Loma Vista
El Segundo, California 90245 (US)

(74) Representative : Wombwell, Francis
Potts, Kerr & Co. 15, Hamilton Square
Birkenhead Merseyside L41 6BR (GB)

(54) Image rendering process and apparatus.

(57) An apparatus for creating a page image which may include complex graphics information for display on a continuous synchronous raster output device by generating bands of graphics information which use an amount of memory which is less than the amount which would be needed to store a complete page image, said apparatus comprising :

a) means for receiving and interpretting commands in a page description language which define the page image to be output ;

b) means for generating a set of graphics orders from the interpretted page description language commands such that :

i) said graphics orders can be processed into bands of bitmap images rich can delivered to the output device at a speed required by the output device ; and

ii) while the graphics information in a first one of said bands is being output by said output device, the bitmap images for a second one of said bands is being generated ;

c) means for generating said bands of bitmap images from said graphics orders ;

d) means for outputting said bands of bitmap images to said output device.

EP 0 470 782 A2

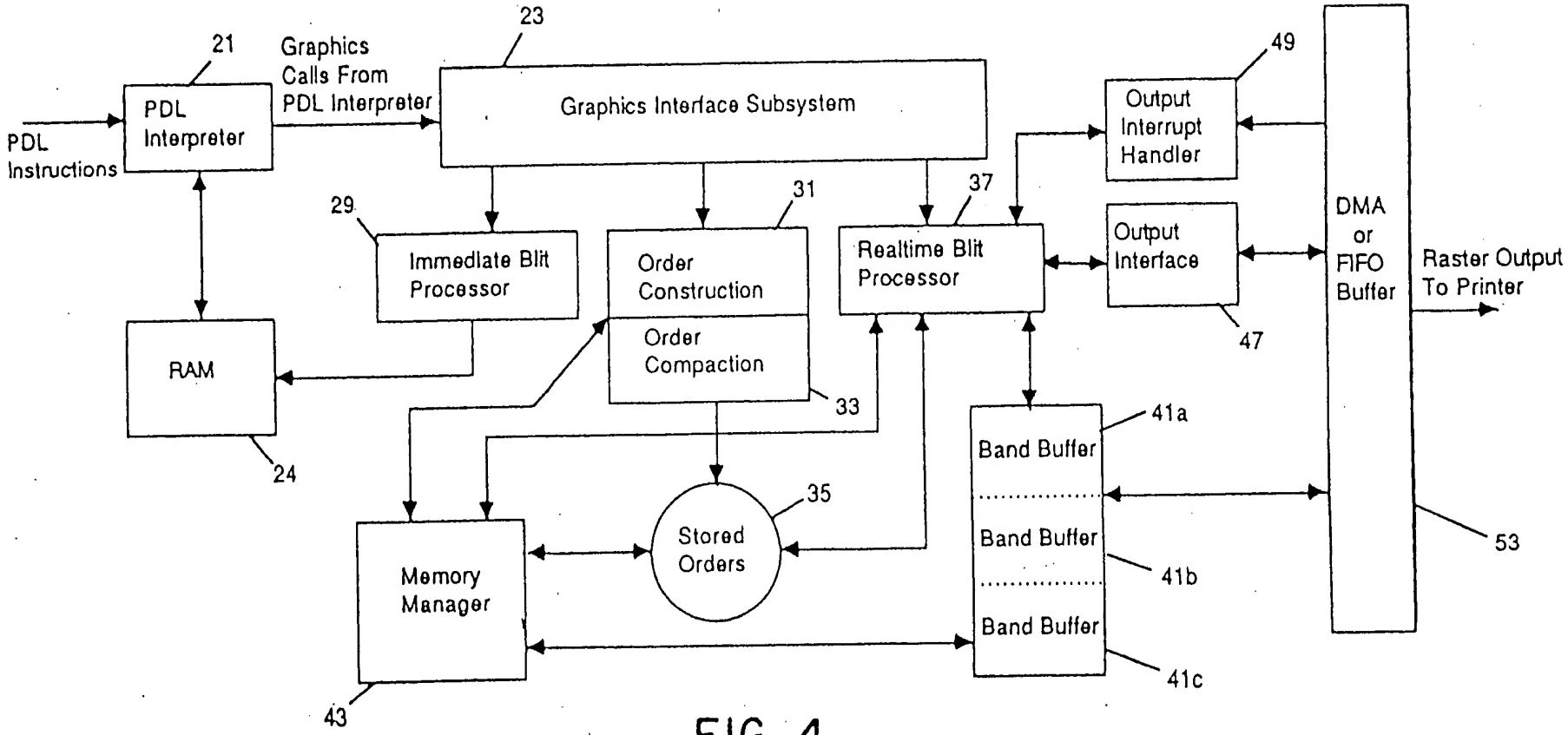
PDL Instructions
21
PDL Interpreter
Graphics Calls From PDL Interpreter
23
Graphics Interface Subsystem
49
Output Interrupt Handler
DMA or FIFO Buffer
Raster Output To Printer
29
Immediate Blit Processor
31
Order Construction
Order Compaction
33
37
Realtime Blit Processor
Output Interface
47
RAM
24
41a
Band Buffer
Band Buffer
41b
Band Buffer
41c
35
Stored Orders
Memory Manager
43
53

FIG. 4

SUMMARY OF THE INVENTION

The present invention is an apparatus and method used to perform image rendering in a manner which is capable of producing complex, high resolution page images for a continuous synchronous raster image output device, such as a laser printer, using a minimum of random access memory.

A continuous synchronous raster image output device requires delivery of the output data in a fixed period of time. If the data is not delivered within the fixed period of time, the output page image is corrupted. The invented method and apparatus decomposes basic graphics functions into a compact series of orders or commands suitable for realtime processing and then generates the output page image on the fly. By way of contrast, excepting for print engines with,limited or no graphics capability, prior art techniques for continuous synchronous raster image output devices require that an entire page be constructed in memory before the output page image is sent to the output device.

The invented technique defines the graphic content of a page in much less memory than the rendered image would otherwise require, but in a form that can be processed and delivered to an output device at the speed required by the output device. In order to accomplish this, graphics functions are represented as low level primitives, referred to herein as orders, to guarantee that they can be converted to bitmapped images in realtime. Additionally, the quantity of orders or commands is reduced by eliminating redundant or overlapping graphics functions to minimize the memory used by the orders.

The invention uses a realtime output processor which operates on a small section of the page image at a time. Such a section is called a band. A band consists of a predetermined number of scanlines where each scanline is a series of pixels to be delivered to the output device at the speed required by the output device.

After building a first band, a second band is created while the first band is being printed. Using prior art techniques, and given the present speed of processors used to generate bitmaps of the image to be printed which is sent to the output device, it would not be possible to finish creating a bitmap for the second band before the printer finished printing the first band. That is, since continuous synchronous raster image printers print at a fixed speed, to ensure that a page is printed correctly, prior art techniques require the creation of a bitmap image of the entire page to be printed before any part of the image is passed to the printer.

By using a technique for generating bitmap images which are less than the complete page, and sending only that portion to the printer, while the bitmap image for another portion of the page is being created, the present invention results in a great cost savings by using less memory. Additionally, even if there is sufficient memory to build a complete page, by using the banding technique of the present invention, a second page can be constructed in one band while a first page in another band is being printed, thereby increasing the throughput of the printer.

The present invention accomplishes its goal by deferring the execution of application generated commands that draw pixels in the page image. These commands are converted to graphics orders rather than directly to bitmaps. The graphics orders take up much less memory than corresponding bitmaps, but can be converted to bitmaps for sending to a printer at a speed fast enough to keep up with the printer.

Since a single graphics command may affect more than one band, an order may be processed multiple times. In this connection, order command blocks, which are represented as tabular structures in memory, include information to allow selection of those graphics commands needed for a given band.

More particularly, image rendering according to the present invention uses an order construction step and two image drawing steps. The order construction step is used to build orders for deferred execution. Applications commands for pixel drawing in memory other than the page image result in bitmaps stored in user memory and do not result in stored orders because these commands are processed immediately. The bitmaps produced by immediate blits later may be used by a PDL interpreter as source bitmaps for blits into the page image.

Inputs to the order construction step are presented through a graphics subsystem interface such that each graphics call delivers arguments which the order construction step reduces to a compact command block which may include a "band" number to be used by the image drawing steps to select a subset of commands for each band of the page to be drawn.

The order construction step attempts to eliminate redundant orders by combining multiple inputs into single command blocks thus reducing the number of command blocks and the memory required to contain them.

The first (immediate) image drawing step proceeds in parallel with the generation of orders that draw objects in the page image. objects drawn in this step are stored in the above-mentioned user memory for use as source data for the second drawing step, but the order command blocks for them are released for reuse.

The second (deferred) image drawing step starts when all orders for the page image have been generated. This step may process order commands multiple times, once for each image band affected. Orders that do not affect the current band are skipped.

The first band affected by an order command is identified by the band number included in the command. When an order command is processed, the band number is updated to the next band number if the order command affects multiple bands.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is block overview diagram of a process for generating a printed page based upon an application which generates graphic images which are to be printed.

Figure 2 is a functional block overview diagram of output controller 15 and bit map image generator 17.

Figure 3 is a functional diagram of an implementation of output controller 15 and bit map image generator 17 according to the prior art.

Figure 4 is a functional diagram of an implementation of output controller 15 and bit map image generator 17 according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention handles rendering of a page image and delivery of the image to the print engine of a continuous synchronous raster image output device. Referring to Figure 1, an application 11 according to instructions from a user through an application service interface generates a representation of an image to be printed in a page description language (PDL) 13 such as Postscript which is available from Adobe Corporation. A PDL defines commands which allow complete and precise control of bitmap and halftone images as well as all of the atributes of character fonts including point size, orientation, shade and slant. The application service interface which generates PDL instructions representing the image is the same regardless of underlying hardware and/or software support. For example, assume a user instructs an application program to draw a line from point A corresponding to a bitmap image coordinate of x1,y1 to point B corresponding to a bitmap image coordinate of x2,y2. Such an instruction would generate a series of Postscript instructions (excluding page and line setup instructions) as follows:

```
newpath
x1 y1 moveto
x2 y2 lineto stroke
```

In the simplest case where there is no special hardware and no banding according to the present invention, the PDL instructions are, in effect, directions to a PDL interpreter in output controller 15 which produces the page image. The PDL interpreter interprets the PDL instructions, producing graphics objects and character bitmaps as directed. Only a few of the PDL instructions result in drawing. The remaining instructions control the use and positioning of objects in the page image.

For example, in the Postscript instructions above, the newpath operator causes the PDL interpreter to initialize internal information to control subsequent operations. The moveto operator establishes an initial position within the coordinate system. The lineto operator establishes a new position with an unexpressed or "unlinked" line between the initial position and the new position. When the interpreter processes the stroke operator, it will generate one or more graphics commands to underlying pixel drawing routines which draw the line in the page image memory.

In this case, the graphics calls are handled by the output controller synchronously. That is, a particular graphics call generates a portion of an image which is to be printed which is stored in a page buffer after which control is returned to the PDL interpreter. The interpretation and drawing steps continue work on the same page image until a page termination operator is found. Then the page image memory is copied to the output device by the hardware. The output copying step may operate asynchronously to the interpreter and graphics functions if there is enough memory to allow the interpreter to produce an image for the next page while the hardware delivers output from the previous page via DMA or other means.

Output controller 15 utilizes a graphics interface which includes graphics support routines which operate in the coordinate system of the output device 19 wherein the origin of the image coordinate system of the output device corresponds to the first pixel written to the output device. The x-axis of the image coordinate system is parallel to the output device scan (raster) lines. Thus, in left to right printing, pixels are output in increasing x-coordinate order, but in top to bottom printing, scan lines are output in increasing y-coordinate order.

The PDL interpreter part of the output controller specifies the size of the page image by giving its width and length. The PDL interpreter part of the output controller also specifies the placement of the image within the physical page by supplying top and left margin parameters. These parameters are expressed in pixels.

The graphics interface of output controller 15 creates two types of calls as follows:

1) calls relating to pages; and

2) calls which draw bit images.

Calls relating to pages are as follows:

1) New Page - performs initialization for a new page image.

2) End Page - starts the realtime printing process if the output device is idle or queues the image orders if the output device is busy.

3) Abort Page - if the printing phase has not started, throws away the orders for the page.

4) Abort All - throws away all pages waiting to be printed.

There are six classes of bit image drawing calls as follows:

1) Two Operand Blit - performs a two dimensional bit block transfer between two bitmaps, a source and a destination.

2) Three Operand Blit - performs a two dimensional bit block transfer using three bitmaps, a source, a halftone, and a destination.

3) Draw Scanline - writes pixel groups on a contiguous block of scanlines within the destination bitmap. The input includes a table defining pixel groups to be written.

4) Draw Scanline Halftone - writes pixel groups to the destination bitmap according to the contents of a halftone source bitmap and a table defining pixel groups to be written.

5) Draw Pixels - draws individual pixels in the destination bitmap according to a pixel table.

6) Draw Pixels Halftone - draws individual pixels in the destination bitmap according to the contents of a halftone source bitmap and a pixel table.

Generally each of the six classes has at least two calls, one for writing to the page image bitmap which the graphics subsystem interface manages; the other for a PDL interpreter managed destination, typically, a RAM user memory.

Interface to Bit Image Drawings Calls

Generally, bitmap image drawing calls expect user and halftone bitmap descriptors as input. A user bitmap descriptor contains bitmap origin byte address, i.e., the memory address of the upper left corner of the bitmap, bitmap width in pixels, i.e., the number of pixels per scanline, and bitmap height in pixels, i.e., the number of scanlines. Bitmap width and height may be arbitrary pixel values. Clipping occurs to the specified bitmap width and height for destination bitmaps; no source bitmap clipping is performed.

While user bitmap dimensions have no boundary constraints, a halftone bitmap width is constrained to be 0 mod 32 pixels (32-bit word multiple). In addition to bitmap origin, width, and height, a halftone bitmap descriptor has three additional fields. Two of the additional fields are x,y origin parameters specifying the point in the halftone bitmap that corresponds to the destination bitmap origin. The third additional field provides halftone bitmap size in bytes. Halftone bitmap usage wraps instead of clipping. This causes the halftone bitmap to appear to repeat in the destination bitmap in both the x and y dimensions.

Normally the contents of a source bitmap do not change except by means of subsequent calls to the graphics support routines performed by output controller 15. However, occasionally the PDL interpreter needs to directly change the contents of a source bitmap after using it in a graphics service call. Since graphics services are not necessarily performed synchronous to the requests (that is, completed prior to the service requests return), a direct change may change the bitmap contents prior to its use by the output controller. In such a case, the bitmap must be declared "volatile," for example, by the PDL interpreter passing a "volatile" flag as an argument to the graphics interface of output controller 15.

This feature must be used sparingly and only where truly . needed due to the expense in memory and processor time. It is not intended as a substitute for PDL interpreter management of required memory. When the PDL interpreter and the other portions of output controller 15 operate synchronously, no problem arises. If the two processes run asynchronously as they do when the controller includes hardware assistance or when double buffering is in effect, the output controller must make a copy of a "volatile" bitmap so that the bitmap does not change before it can be used. The caller, i.e., the PDL interpreter can flag a bitmap as volatile in its descriptor.

Figure 2 shows the functions according to the present invention performed by output controller 15 which inputs PDL commands, interprets the PDL commands and generates graphics commands or orders, and by bitmap image generator 17 which generates a bitmap of the image to be printed from the graphics orders and outputs the image to the printer.

A functional block diagram of output controller 15 and bit map image generator 17 as implemented in the prior art will now be described with reference to Figure 3. Specifically, output controller 15 and bitmap image generator 17 comprise a graphics interface subsystem 23 which receives the graphics calls generated by PDL interpreter 21 based upon the PDL instructions 13 generated by application 11. The graphics interface subsystem passes the graphics calls to immediate blit processor 29. User memory 24 is a random access memory

used by PDL interpreter 21 as a temporary storage of source bitmaps which will be passed to graphics interface subsystem 23 as source arguments to graphics calls. Memory manager 43 allocates and deallocates RAM memory blocks as requested by graphics interface subsystem 23. When an End Page instruction is called by the PDL interpreter, the actual bitmap images to be printed are transferred from page buffer 42 to DMA hardware or a FIFO buffer 53 for transfer to a printer by operation of output interface 47, output interrupt handler 49, and graphics interface subsystem 23.

The specific manner in which the elements shown in Figure 3 operate to produce a raster output suitable to be sent to a printer is as follows.

PDL interpreter 21 processes the incoming PDL source statements, calling the graphics interface subsystem 23 as necessary using calls relating to pages and bit image drawing.

When the PDL interpreter calls the graphics interface subsystem 23 to start a new page, the graphics interface subsystem calls memory manager 43 to allocate memory for the page image.

Subsequently, the graphics interface subsystem 23 calls immediate blit processor 29 for each blit command from the PDL interpreter 21 after checking the command for errors and adjusting the parameters as necessary to perform destination clipping, i.e., assuring that a blit cannot extend beyond the boundaries of the destination as defined in the blit command arguments or the extent of the page image.

PDL interpreter 21 calls the graphics interface subsystem 23 when it has completed processing for the page. The graphics interface subsystem then calls output interface 47 to deliver the image for output.

The output interface checks to see if output is in progress and if not starts the printer hardware and delivers the first data to the DMA or FIFO device 53.

If output is in progress, output interface 47 places the new output image on its internal queue to be started when current activity completes.

Interrupt handler 49 operates by continuing the delivery of data to the DMA or FIFO 53 and by starting a queued image, if any, when the output for the current image is complete. When the output of a page image is completed, interrupt handler 49 calls a completion routine in graphics interface subsystem 23 to announce this completion. At this time graphics interface subsystem 23 can return the page image buffer 42 to memory manager 43 or re-use it for another page.

Figure 4 is a functional block diagram showing the functions of output controller 15 and bitmap image generator 17 when implemented according to the teachings of the present invention. Before describing the elements of Figure 4 in detail, it will be helpful to explain the major differences between the technique for rendering the output image in small bands according to the present invention and rendering it in a full page image buffer according to the prior art. These differences are as follows:

1. When using a full page buffer, graphics interface subsystem 23 uses the immediate blit processor for every pixel drawing call the PDL interpreter makes. That is order construction 31, order compaction 33, realtime blit processor 37, and the stored graphics orders 35 shown in Figure 4 are not used in the prior art.
2. Additionally, in the prior art, output interface 47 and output interrupt handler 49 do not need to coordinate processing with a realtime blit processor which does not exist using prior art techniques, eliminating the need for what may be called an "extended interrupt processing thread."

The invented banding technique includes the following requirements:

1. An ability to differentiate between graphics requests which can be performed immediately by immediate blit processor 29 and those which must be converted to stored orders for subsequent handling by realtime blit processor 37.
2. Design of the stored order formats.
3. An order compaction function.
4. A capability for updating the stored orders by the realtime blit processor to eliminate those that it completes or mark those that it must reprocess in a way so it can efficiently reprocess an order that spans multiple bands.

While the invented technique may be implemented in software, when implemented using co-processor hardware, the hardware replaces both the immediate blit processor and the realtime blit processor. In addition, in the preferred embodiment, a hardware implementation incorporates the output FIFO or DMA device 53. A hardware implementation of the realtime blit processor does the same type of order updating as that performed by a software implementation. However, since a hardware implementation runs much faster than a software implementation, the hardware implementation can band much more complex pages and keep up with much faster output devices. It also may employ a separate hardware bus for access to the band buffers 41a, 41b and 41c.

However, in a hardware implementation, software will still normally be used to construct the stored orders since the order information must be represented in a manner suitable for processing by a state machine which

can be implemented in a cost effective manner.

The following is a brief description of each of the functional elements comprising the invention as shown in Figure 4.

Graphics Interface Subsystem 23

PDL interpreter 21 calls graphics interface subsystem 23 which is a set of standard interface routines for graphics services. The graphics interface determines whether it can perform an immediate blit based on whether or not the destination is user memory 24 or band buffers 41. If it cannot do an immediate blit, it calls order construction function 31 to create stored orders 35.

After either the immediate blit is done or the orders have been constructed, the graphics interface subsystem returns to PDL interpreter 21.

Immediate Blit Processor 29

This function draws pixels in user memory 24 as requested and returns. This function is known in the prior art and is performed whenever pixel drawing commands affect a user memory destination.

Order Construction 31

Order construction processing 31 generates orders 35 which are stored in a memory for subsequent use during realtime blit processing performed by realtime blit processor 37. For certain types of orders where it may be possible to combine subsequent orders with ones already known, it calls the order compaction function 33 to keep track of the orders. In the preferred embodiment, at least scanline blit orders are subject to compaction. Other types of orders can be compacted in accordance with the teachings of this invention.

Order Compaction 33

This function keeps records of the orders that may be compressed until informed by the order construction processing function that it must wrap up a set of orders. Then it reinitializes its internal information to start collecting information about the next set. This function is very important in cases where many small scanline blit orders operate on a small region of the page image. Such compaction provides major reductions in the amount of memory required in cases because the blits overlap.

Stored Orders 35

Stored orders 35 is an internal representation of commands to create the images the application requested.

Realtime Blit Processor 37

When the PDL interpreter issues a call to say it has completed all the requests for a page, graphics interface subsystem 23 starts the realtime processing phase. It traverses stored orders 35 once for each band to be output. When all orders that effect a band have been processed, it calls output interface 47 to queue the band for output or start it immediately if the DMA or FIFO buffer output hardware 53 is available. If another band buffer 41a, 41b or 41c is available, it will construct the next band. When no more buffers are available, it returns to the graphics interface subsystem 23 which returns to PDL interpreter 21. The PDL interpreter may continue interpreting input and working on the next page. From this point on, realtime blit processor 37 is driven by interrupts inititated by output hardware 53.

Realtime blit processor 37 converts the stored orders into a bitmap image corresponding to a band or section of the page to be printed which is stored in band buffers 41a, 41b or 41c.

Specifically, interrupt handler 49 is notified by output hardware 53 when a band has been delivered to the printer. Realtime blit processor 37 is then called as an "extended interrupt thread" to continue its processing.

Band Buffers 41a, 41b, 41c

Band buffers 41a, 41b and 41c is a random access memory used to hold the bitmapped page image. If banding is not used, a single band buffer is used to hold a complete page. Otherwise there may be 2 or 3 band buffers of a size much smaller than the complete page.

Memory Manager 43

Memory manager 43 allocates and deallocates blocks of memory in response to calls from graphics interface subsystem 23, order construction 31, order compaction 33 or realtime blit processor 37. Memory manager 43 is a software function that manages the memory containing the stored orders and the band buffers according to well known memory management techniques which need to be employed when the other software components need to acquire or free blocks of memory. The actual bitmap images to be printed are transferred from the band buffers 41 to DMA hardware or a FIFO buffer 53 for transfer to a printer by operation of output interface 47, output interrupt handler 49, realtime Blit processing 37 and memory manager 43. In this connection, although three band buffers are shown in Figure 4, in practice, there may be two, three or more band buffers, although in most cases two or three band buffers are sufficient.

Output Interface 47

Output interface 47 receives completed bands from the realtime blit processor and either passes them on to the output hardware 53 or queues them for output interrupt handler 49 to pass to the output hardware when it is no longer busy. With the help of output interrupt handler 49, output interface 47 monitors synchronization signals from the printer engine to determine when it can deliver a band for output. Print engines send a "frame sync" signal to indicate top of page. The output interface cannot deliver any data until the "frame sync" signal is true.

When a FIFO buffer is used as the output hardware, the software moves data from the band buffer to the FIFO a scanline at a time. A DMA controller can be used the same way. Top and left margins are created by delivering zeroes for an output device that writes black or by delivering ones for an output device that writes white.

Output Interrupt Handler 49

As the output hardware finishes delivering a band to a video interface of the print engine, it generates an interrupt. The output interrupt handler gets the next band from its queue and passes it to the output hardware. It then returns the previously completed band buffer so the realtime blit processor may reuse it.

When the output interrupt handler returns, realtime blit processor 37 runs again until it has rendered another band before the graphics interface system returns to the PDL interpreter. This "extended interrupt thread" is implemented differently on different CPUs depending on their architectures. The "extended interrupt thread" runs on a separate stack with CPU priority lowered so that it can be interrupted.

Output Hardware 53

When banding according to the present invention is implemented in software, a FIFO or DMA controller is used to handle video output to the print engine. If the banding implementation is done hardware, it may incorporate a FIFO or DMA controller output hardware.

In the preferred embodiment, the functional elements shown in Figure 4 are implemented in hardware, although if one or more of such elements are implemented in software, the function performed by each such element is actually performed by operation of a microprocessor executing a set of instructions in a memory such that the microprocessor generates control signals appropriate to the function being performed. In this connection, just as the specific hardware implentation details are not needed for a person skilled in the field of the invention to make and use the same, the particulars of a microprocessor, memory, data and address bus, clock and the like needed to implement the invention in software would be readily apparent to a person skilled in the art based upon the description provided herein.

In the preferred embodiment the various elements of Figure 4 comprising the present invention are implemented as follows.

Graphics interface subsystem 23

All service requests from the PDL interpreter enter the graphics interface module. These service requests are: New Page, End Page, All Blits With User Destinations, and All Blits With Page Image Destinations.

New Page

```
The New Page call performs initialization for a new page
   Determine page image memory requirements from input arguments
   Call the Memory Manager to allocate band buffers for the page
   image
      If Memory Manager cannot supply required memory
         Then if application passed NO_PGWAIT flag or no output in
         progress
               Return with error
         Wait for memory to become available
   Initialize memory and control structure for page
   Return with no error
```

End Page

End Page starts image printing if the page is not banded or starts the Realtime Blit if the image is banded.

```
   Save the address of the application completion function
   Set initial parameters such as current band number and copy
   number in the control structure for this page
   If the page is not banded
      Call Output Interface to deliver the image buffer for output
      Return
   Else
      Call the Realtime Blit routine, Start Next Page
      Return
```

All Blits With User Destinations

```
   If the page is not banded or the application called with the
   DO_ONCE flag
      Call Immediate Blit to perform the pixel drawing service
      Return
   Else
      Call Order Construction to create Stored Order for the blit
      Return
```

All Blits With Page Image Destinations

```
If the page is not banded
    Call Immediate Blit to perform the pixel drawing service
    Return
Else
    Call Order Construction to create Stored Order for the
blit
    Return
```

Immediate Blit Processor 29

The immediate blit function draws pixels in memory. Each type of blit has a set of arguments that determine the size of the image to be drawn, its location in memory, and the particular blit function.

Two Operand Blit

The two operand blit performs a two dimensional bit block transfer operation between two bitmaps. The source and destination bitmaps are defined by the passed bitmap descriptors.

Three Operand Blit

The Three Operand Blit performs a two dimensional bit block transfer operation using three bitmaps. The destination is a user defined bitmap.Each source, halftime, and destination pixel is combined according to the pixop parameter and then rewritten to the same pixel in the destination bitmap. The third bitmap is a halftone (or inking) bitmap typically used to write an arbitrary pattern into the destination frame. The halftone bitmap and its origin in the destination bitmap are specified via halftone parameters. The halftone bitmap is used in a wraparound fashion: pixel use off the end of a scanline continues from the beginning of the same scanline; scanline use off the end of the bitmap continues with the first scanline in the bitmap.

Scanline Blit

Scanline writes pixel groups on a contiguous block of scanlines within a destination bitmap. The user passes a table defining the pixel groups. A pixel group consists of a contiguous sequence of pixels on a single scanline. The table structure allows for multiple groups per scanline.

Draw Scanline Halftone

Draw Scanline Halftone writes pixel groups to the destination bitmap according to the contents of the halftone bitmap and the pixop boolean. The user passes a halftone bitmap descriptor as well as a destination bitmap descriptor and a table defining the pixel groups. A pixel group consists of a contiguous sequence of pixels on a single scanline. The table structure allows for multiple groups per scanline and for the skipping of scanlines.

Draw Vertical Line

Draw Vertical Line performs a service analogous to Draw Scanline except that the pixel groups represent a vertical line or lines. The user passes a table defining the pixel groups. A pixel group consists of a contiguous sequence of pixels. The table structure allows for multiple pixel groups.

Draw Pixels

Draw Pixels processes the specified pixel table, writing the affected pixels according to the destination only boolean, pixop. The destination is defined by the destination bitmap descriptor.

The pixel table consists of the number of xy pairs.

Draw Pixels Halftone

Draw Pixels Halftone processes the specified pixel table, writing the affected pixels according to the halftone bitmap and the boolean, pixop. The destination is defined by the destination bitmap descriptor. The pixel table consists of the number of xy pairs specified in the count parameter.

Logical Pixel Operations

The value of each pixel written is determined by a logical combination of the pixel(s) involved in a particular operation. Some operations such as scanline involve only a destination pixel at each coordinate. Others such as 2blit involve a source and destination pixel at each coordinate while yet others such as 3blit involve halftone, source, and destination pixels. There are 256 possible combinations of three pixels and the four boolean operators: NOT, AND, OR, XOR. All such combinations can be expressed using bitwise logical combinations of the following definitions:

Logical Pixel Operations

The value of each pixel written is determined by a logical combination of the pixel(s) involved in a particular operation. Some operations such as scanline involve only a destination pixel at each coordinate. Others such as 2blit involve a source and destination pixel at each coordinate while yet others such as 3blit involve halftone, source, and destination pixels.

There are 256 possible combinations of three pixels and the four boolean operators: NOT, AND, OR, XOR. All such combinations can be expressed using bitwise logical combinations of the following definitions:

GRS_DST = 10101010
GRS_SRC = 11001100
GRS_HT = 11110000
GRS_WHITE = 00000000
GRS_BLACK = 11111111

The WHITE and BLACK are used when the written pixels are to have the corresponding value without regard to prior pixel values. The other three values can be combined to produce all other possible unique boolean combinations of three pixels with four operators. Not all combinations necessarily produce useful effects. For example:

to copy source pixels to destination...
pixop = GRS_SRC 11001100

to complement destination pixels ...
pixop = NOT GRS_DST 01010101

to superimpose a source bitmap onto the destination bitmap ...
pixop = GRS_SRC OR GRS_DST 11101110

to superimpose a negative of a source bitmap onto the destination bitmap ...
pixop = (NOT GRS_SRC) OR GRS_DST 10111011

to white destination pixels corresponding to an image in a source bitmap ...
pixop = (NOT GRS_SRC) AND GRS_DST 00100010

to halftone all destination pixels which are black in the source bitmap leaviny other destination pixels unchanged ...
pixop = (GRS_SRC AND GRS_HT) OR ((NOT GRS_SRC) AND GRS_DST) 11100010

```
All immediate blits proceed as follows:
   Check user arguments
   If argument error
      Return error
   If clipping required
      Adjust parameters for low level blit call
   If blit hardware available
      Set up hardware command/order
      If hardware busy
         Add order to queue for start by interrupt handler
         Return
   Else
      Call appropriate low level blit function
   Return
```

Order Construction 31

Order construction builds stored orders for the realtime blit software for hardware to use in building the page image. Each stored order includes the information needed by the immediate blit routines and information used by the realtime blit routines such as the first page image band affected by the order.

```
Build a stored order from the blit arguments passed
If the application called with the VOLATILE flag for the source
bitmap
   Call the memory manager to get space to copy the bitmap
   If there is not enough memory available
      Return with error code
   Else copy the source bitmap
If the requested blit is not subject to compaction
   Call order compaction terminate data structure routine
Else
   Call order compaction add order routine
Return
```

Order Compaction 33

Order compaction attempts to combine multiple orders into a single order, eliminating redundancy for multiple orders operating on the same destination pixel groups. The compaction routine keeps track of pixel groups

in an internal data structure. It continues adding to the structure until Order Construction calls with a request to terminate the current set. When this happens, the current data structure is converted to an order and placed in order storage. The next call to order compaction begins a new data structure. A number of commonly used data structures can be used to keep track of the orders subject to compaction, including a linked list, an array, or a binary tree. If a fixed length, non-extensible structure such as an array is used, the size of the array limits the number of orders which may be combined. This does not defeat the compaction scheme; If the data structure becomes full, the Add Orders function can call the Terminate routine to terminate the current set and start a new one. This may reduce the level of compaction somewhat.

Add Order

```
If there is no current data structure
   Call Memory Manager to allocate space for data structure
      If error
         Return error
   Initialize new data structure
   Place input pixel groups in the new data structure
   Return
Set "current input group" to the first pixel group in the input
Set "current existing group" to the first pixel group in the
existing data structure
While a "current input group" remains to process ("current
input group" less than or equal to number of input pixel groups)
{
   Compare "current input group" y position with "current
   existing group" y position
   If equal
      Compare starting x (horizontal) position of
      "current input group" with x of "current existing
      group"
      If "input" x_start greater than or equal to "existing"
```

```
        x_start and "input" x_end less than or equal to
        "existing" x_end
              Alter "existing" pixel group to include "input"
              pixel group
              Increment "current input group"
      Increment "current existing group"
      If "current existing group" greater than last existing
      group (no more existing groups)
              Add remaining input groups
              Return
    }
    Return
```

Terminate Data Structure

Convert data structure to order
Attach new order to realtime orders
Call Memory Manager to deallocate temporary data structure
Return

Realtime Blit Processor 37

The realtime blit routine begins by generating as many bands as it has available band buffers. It then delivers these initial bands to the output interface 47 in order and returns to the caller. When interrupt handler 49 determines that a band has been printed, realtime blit processing is restarted via an "extended interrupt thread" to generate the next band in the available band buffer.

Start Next Page:

```
    If this page has a pre-allocated band buffer
      If Band Generation is busy
        Queue the page control structure
        Return
      Call Band Generation passing the page control structure
        Return
    Else
      Return
```

Band Generation Entry Point

```
While image generation not done
{
   If all image band buffers are busy
   Return
   While an order in the list of Stored Orders has its band
   number equal to the current band number
   {
   Extract information from the order to define the blit
required to draw the indicated pixels or the portion of them that
fit in the current band
   Call the low level blit function
   If the order cannot be completed in the current band
      Update the band number in the order to current band + 1
      Update other parameters as necessary
   Else
```

```
        Mark the order as complete
      }
      If this is the first band of the page
        Set first band flag to pass to Output Interface
        Also set completion routine address
      If this is the last band of the page
        Set last band flag to pass to Output Interface
      Call Output Interface passing output structure
      If last band of image was generated
        If another image copy required
            Decrement copy count
            Reset band number to 1 in current control structure
        Else
            If another image is queued
              Get its control structure from queue
            Else
              Set band generation done
    }
    Return
```

Completion Function: (Called by Interrupt Handler)

```
    Mark band buffer idle
    If more bands in current image and Band Generation is idle
      Start Band Generation via "extended interrupt thread"
    Else
      Call PDL interpreter completion function if there is one
    Return
```

Memory Manager 43

The Memory Manager provides RAM allocation and deallocation services such that the other functions can request and release variable size blocks of memory.

Allocate Block

```
Search internal data structures for a free memory block equal or
greater in size than the size requested
   If a block equal to the size requested is found
      Mark block as in use
      Return to caller with address of block
   Else
      Divide free block into two
      Mark the new block for the caller as in use
      Return to caller with address of block
```

Free Block

```
Verify address of block
If address invalid
   Return with error code
Mark block as free
Return with no error
```

Output Interface 47

The Output Interface receives arguments defining the size of the band buffer, a completion function address, left and top margin information, a count of the number of times to print the band, and flags indicating first and last band. In the non-banded case the band is a full memory image of the page and both first-and last band flags are set.

```
If the output device is busy
   Place band information on internal queue
   Return
Else
   Initialize internal data structures
   Start the output device
   Wait for synchronization signal
   Copy the first scanline or multiple scanlines to FIFO or
start
   DMA device to begin image delivery
   Return
```

Output Interface Handler 49

The hardware interrupts when more data must be delivered to the FIFO or DMA device.

```
Check internal information to determine whether more scanlines
in    the current band buffer must be delivered
   If no more scanlines in current band
      Call Realtime Blit Completion Function
      Dequeue next band
      If no more bands
         Return
         Else
            Initialize internal information for this band
      Copy to FIFO or start DMA
      Return
```

Claims

1. An apparatus for creating a page image which may include complex graphics information for display on a continuous synchronous raster output device by generating bands of graphics information which use an amount of memory which is less than the amount which would be needed to store a complete page image, said apparatus comprising:
   a) means for receiving and interpretting commands in a page description language which define the page image to be output;
   b) means for generating a set of graphics orders from the interpretted page description language commands such that:
      i) said graphics orders can be processed into bands of bitmap images which can delivered to the output device at a speed required by the output device; and
      ii) while the graphics information in a first one of said bands is being output by said output device, the bitmap images for a second one of said bands is being generated;
   c) means for generating said bands of bitmap images from said graphics orders;
   d) means for outputting said bands of bitmap images to said output device.

2. The apparatus defined by Claim 1 wherein said graphics orders generation means comprises:
   a) graphics interface subsystem means for receiving said interpretted page description language commands and passing said received commands to at least one of a memory manager means, an immediate blit processor means and an order construction means;
   b) said memory manager means for allocating and deallocating blocks of a memory in response to calls from said graphics interface subsystem means, said order construction means and an order compaction means;
   c) said immediate blit processor means for drawing pixels in a user memory based upon pixel drawing commands passed by said graphics interface subsystem means;
   d) said order construction means for building stored orders based upon blit arguments passed by said graphics interface subsystem means;
   e) said order compaction means for combining multiple orders operating on the same destination pixel groups built by said order construction means.

3. The apparatus defined by Claim 2 wherein said band generation means comprises a realtime blit processor means for traversing said stored orders once for each band to be output and converting the stored orders into bitmap images corresponding to sections of said page image to be output and storing said bitmap images for each said band into a corresponding band buffer.

4. The apparatus defined by Claim 1 wherein said band outputting means comprises:
   a) output interface means for receiving completed bands of bitmap images from said realtime blit processor means and passing said bands to an output buffer means based upon signals received from said output buffer means and an output interrupt handler means;
   b) said output interrupt handler means for retreiving a band containing said bitmap images from a queue and passing said band to said output buffer means;
   c) said output buffer means for sending said bitmap images to said output device.

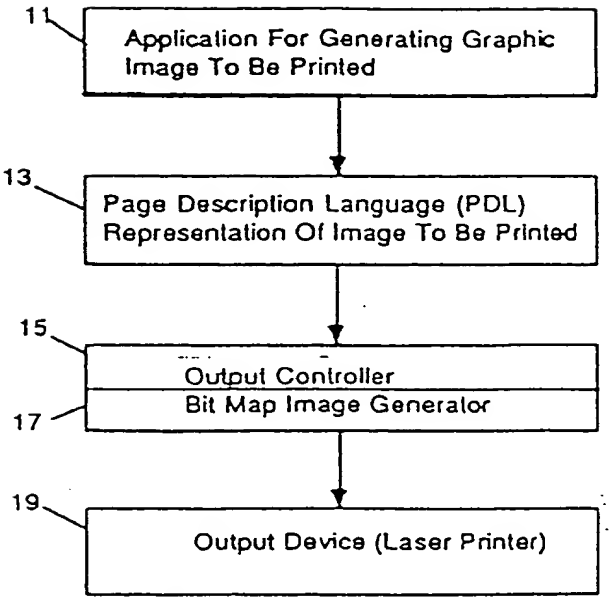
11
Application For Generating Graphic Image To Be Printed
13
Page Description Language (PDL) Representation Of Image To Be Printed
15
Output Controller
17
Bit Map Image Generator
19
Output Device (Laser Printer)

FIG. 1

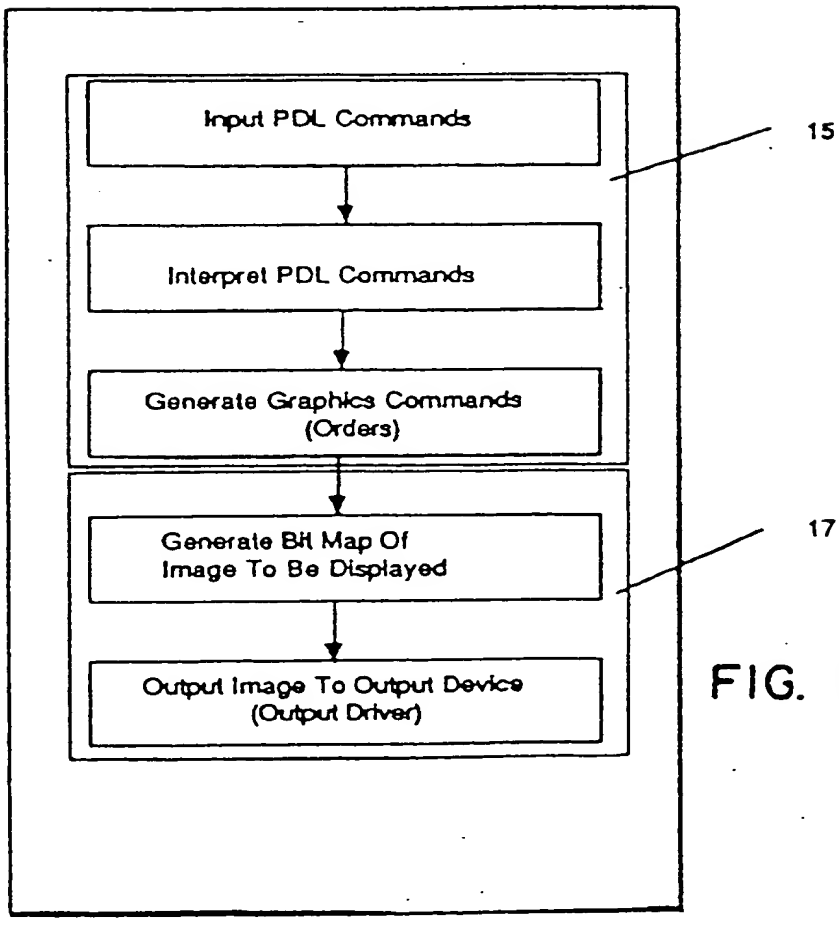
Input PDL Commands
Interpret PDL Commands
Generate Graphics Commands (Orders)
15
Generate Bit Map Of Image To Be Displayed
Output Image To Output Device (Output Driver)
17

FIG. 2

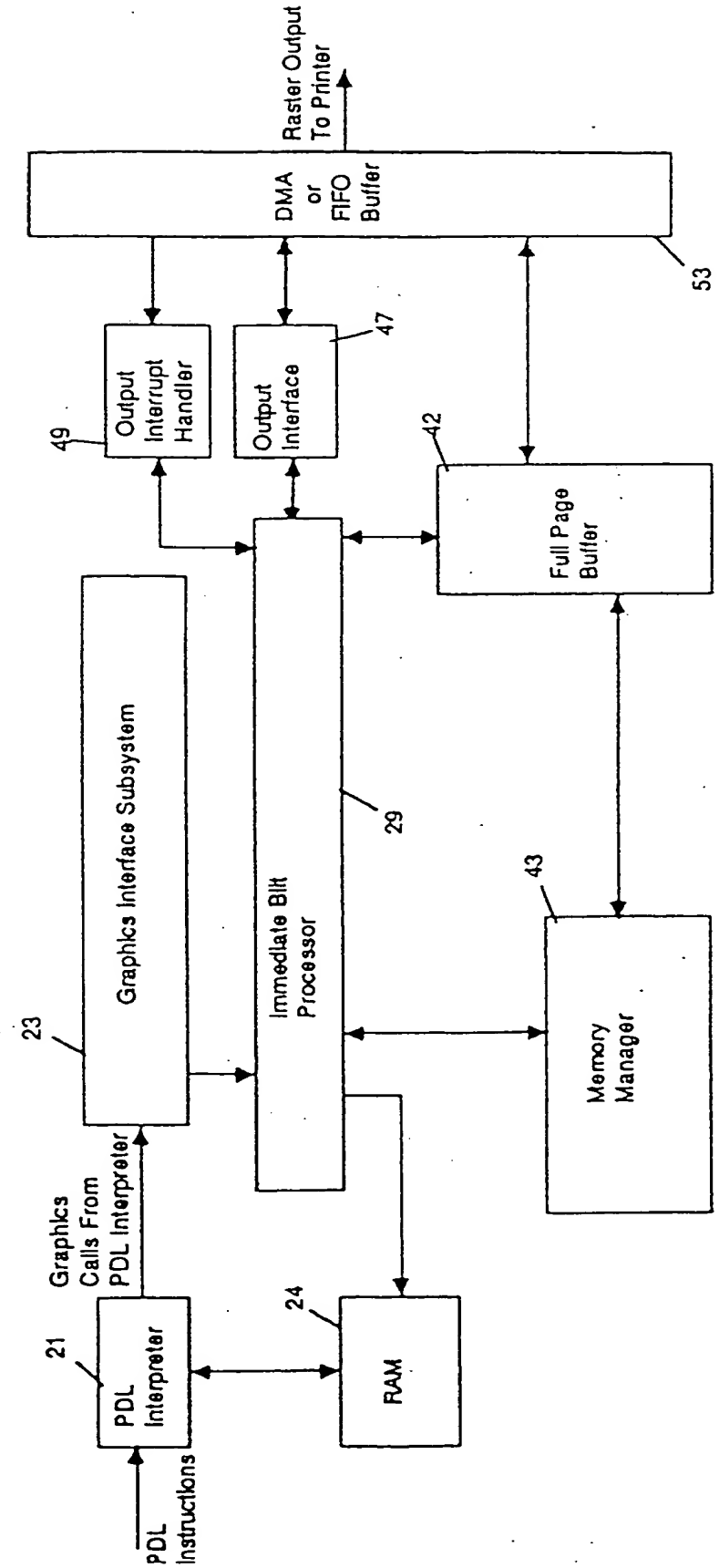

FIG. 3

PRIOR ART

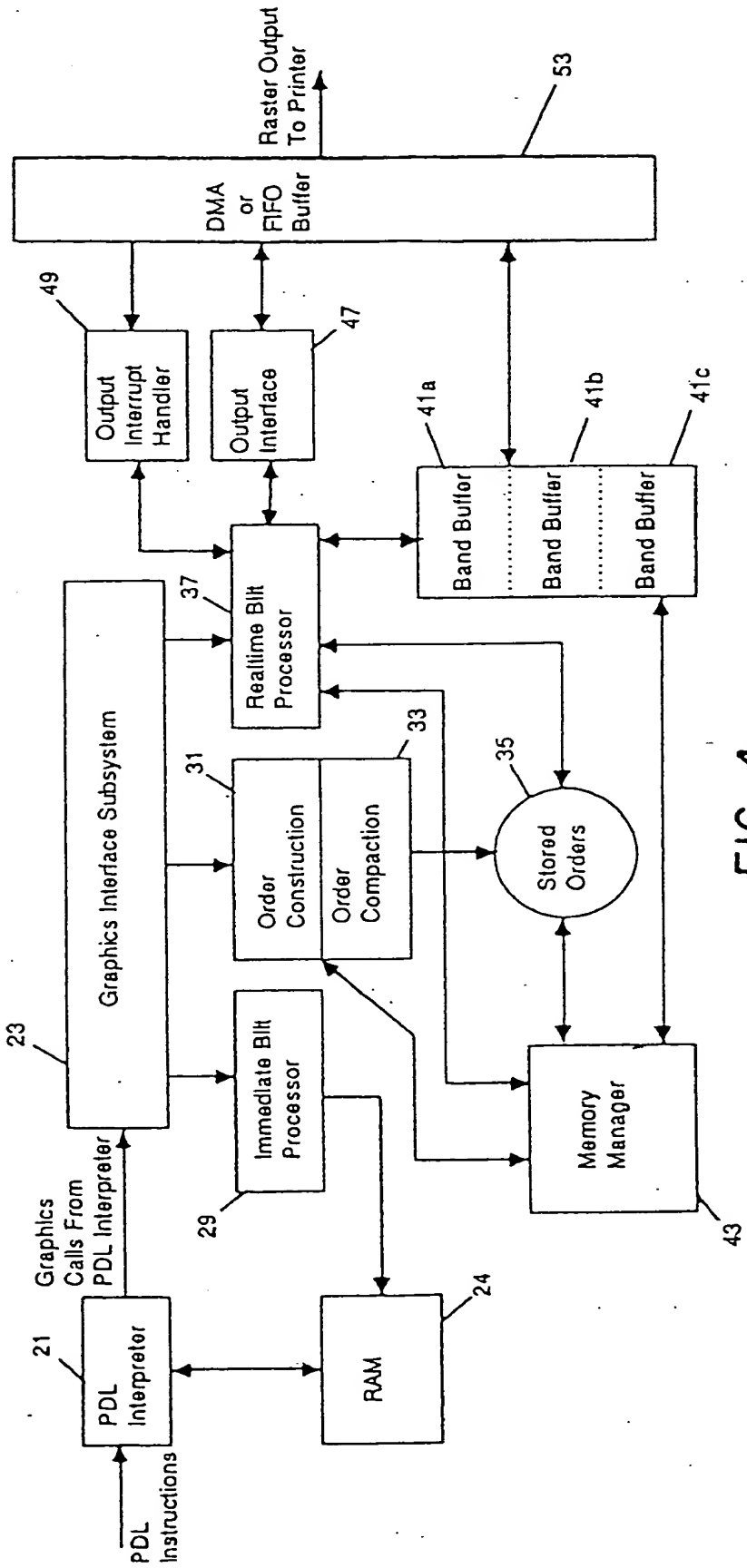
Raster Output To Printer
53
DMA or FIFO Buffer
49
Output Interrupt Handler
47
Output Interface
41a
41b
41c
Band Buffer
Band Buffer
Band Buffer
37
Realtime Blit Processor
Graphics Interface Subsystem
31
Order Construction
33
Order Compaction
35
Stored Orders
23
Immediate Blit Processor
29
Memory Manager
43
Graphics Calls From PDL Interpreter
21
PDL Interpreter
24
RAM
PDL Instructions

FIG. 4